

说明：本视频是对王道书 4.1.5、4.1.6 的总结，不对应书中任何内容

文件的逻辑结构（4.1.5）、文件的物理结构（4.1.6）是非常容易混淆的两个知识点，本视频是为了帮助初学者捋清二者关系而制作的。

建议：学完本视频后，可以再快速过一遍 王道书 4.1.5、4.1.6，巩固理解

本节内容

逻辑结构 Vs 物理结构

关注公众号【研途小时】获取后续课程完整更新！

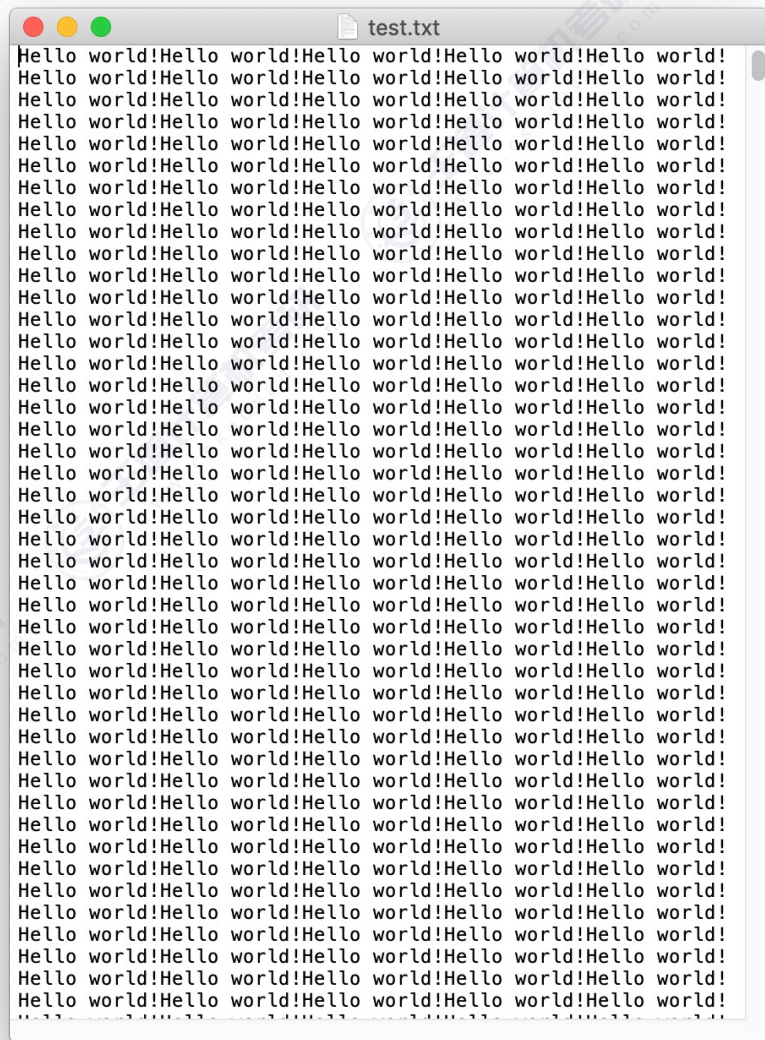
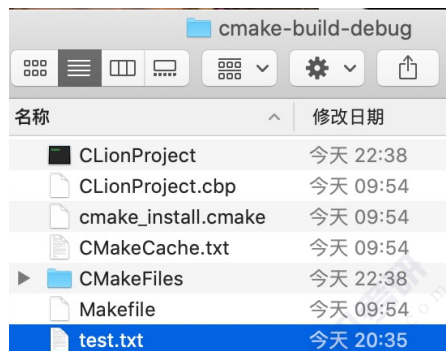
王道考研/CSKAOYAN.COM

傻傻分不清楚？



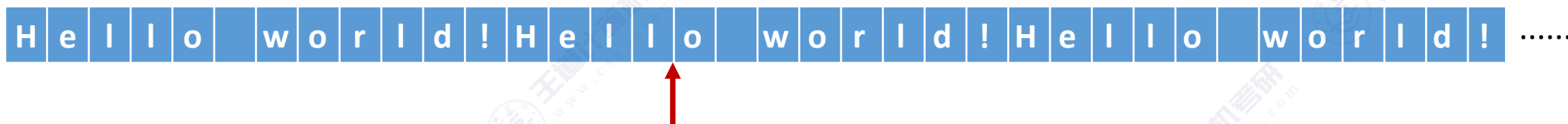
例：C语言创建无结构文件

```
FILE *fp = fopen("test.txt", "w"); //打开文件
if( fp == NULL ){
    printf("打开文件失败!");
    exit(0);
}
//写入1w个Hello world
for (int i=0; i<10000; i++)
    fputs("Hello world!", fp);
fclose(fp); //关闭文件
```



逻辑结构（从用户视角看）

每个字符1B。在用户看来，整个文件占用一片连续的逻辑地址空间



Eg: 你要找到第16个字符（编号从0开始）

```
FILE *fp = fopen("test.txt", "r");    //以"读"方式打开文件
if( fp == NULL ){
    puts("Fail to open file!");
    exit(0);
}
fseek(fp, 16, SEEK_SET);               //读写指针指向16
char c = fgetc(fp);                    //从读写指针所指位置读出1个字符
printf("字符: %c", c);                 //打印从文件读出的字符
fclose(fp);                            //关闭文件
```

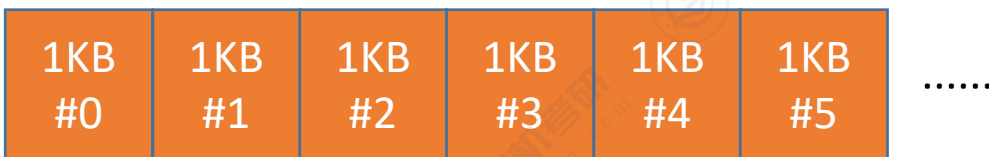
用户用逻辑地址访问文件

```
Run: CLionProject x
"/Users/honglin/L
字符: o
Process finished
```

物理结构（从操作系统视角看）

H e l l o w o r l d ! H e l l o w o r l d ! H e l l o w o r l d !

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



被操作系统拆分为若干个块，逻辑块号相邻

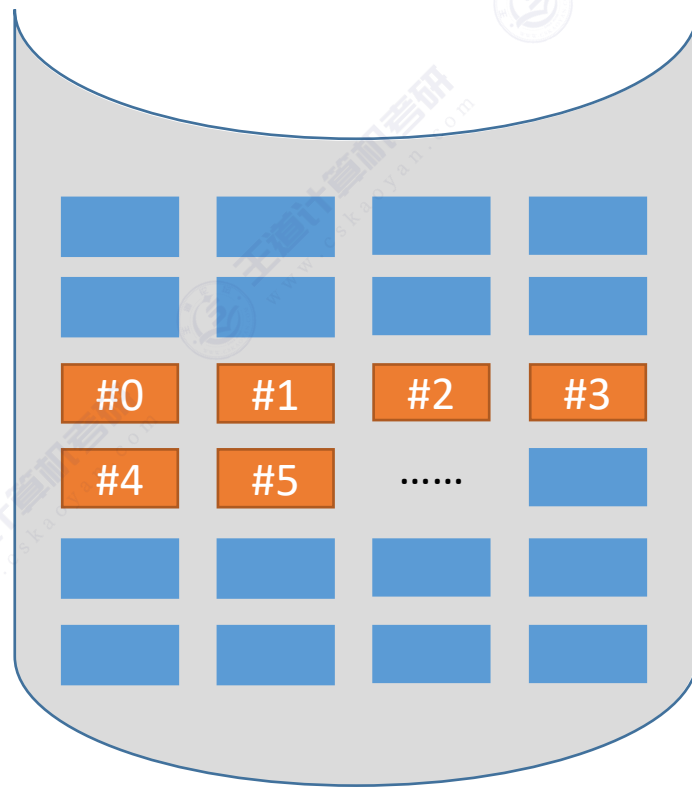
用户：

使用 C语言库函数 `fseek`，将文件读写指针指向位置 `n`

使用 C语言库函数 `fgetc`，从读写指针所指位置读出 1B 内容

指明逻辑地址

`fgetc` 底层使用了 `Read` 系统调用，操作系统将（逻辑块号，块内偏移量）转换为（物理块号，块内偏移量）

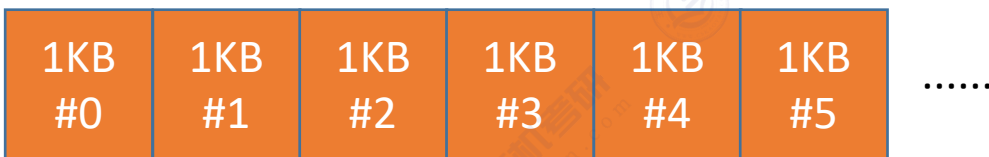


连续分配：逻辑上相邻的块物理上也相邻

物理结构（从操作系统视角看）

H e l l o w o r l d ! H e l l o w o r l d ! H e l l o w o r l d !

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



被操作系统拆分为若干个块，逻辑块号相邻

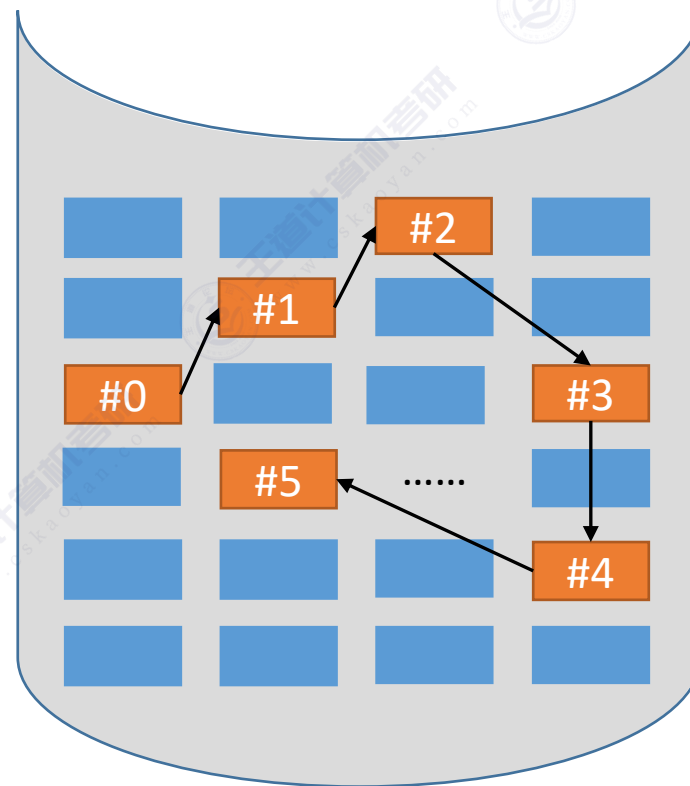
用户：

使用 C语言库函数 `fseek`，将文件读写指针指向位置 `n`

使用 C语言库函数 `fgetc`，从读写指针所指位置读出 1B 内容

指明逻辑地址

`fgetc` 底层使用了 `Read` 系统调用，操作系统将（逻辑块号，块内偏移量）转换为（物理块号，块内偏移量）

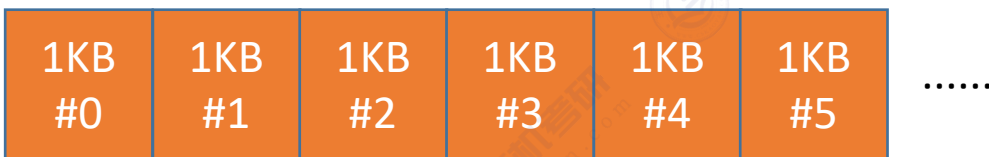


链接分配：逻辑上相邻的块在物理上用链接指针表示先后关系

物理结构（从操作系统视角看）

H e l l o w o r l d ! H e l l o w o r l d ! H e l l o w o r l d !

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



被操作系统拆分为若干个块，逻辑块号相邻

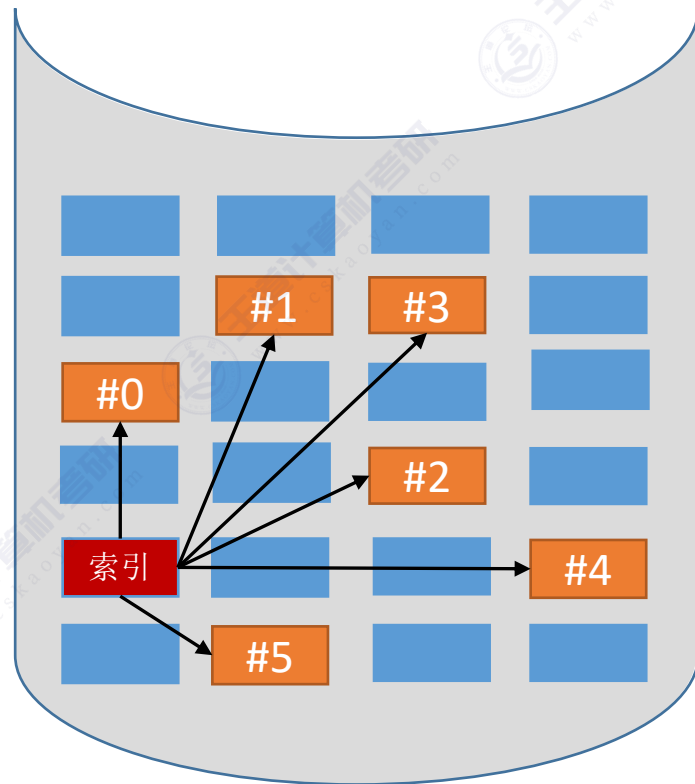
用户：

使用 C语言库函数 `fseek`，将文件读写指针指向位置 `n`

使用 C语言库函数 `fgetc`，从读写指针所指位置读出 1B 内容

指明逻辑地址

`fgetc` 底层使用了 `Read` 系统调用，操作系统将（逻辑块号，块内偏移量）转换为（物理块号，块内偏移量）



索引分配：操作系统为每个文件维护一张索引表，其中记录了逻辑块号→物理块号的映射关系

例：C语言创建顺序文件

```
typedef struct {  
    int number;           //学号  
    char name[30];        //姓名  
    char major[30];       //专业  
} Student_info;
```

//以"写"方式打开文件

```
FILE *fp = fopen("students.info", "w");
```

```
if(fp == NULL) {  
    printf("打开文件失败!");  
    exit(0);  
}
```

```
Student_info student[N]; //用数组保存N个学生信息
```

```
for(int i = 0; i < N; i++) { //生成 N 个学生信息
```

```
    student[i].number = i;  
    student[i].name[0] = '?';  
    student[i].major[0] = '?';  
}
```

//将 N 个学生的信息写入文件

```
fwrite(student, sizeof(Student_info), N, fp);
```

```
fclose(fp);
```

用户视角：
每个学生记录占 64B
sizeof(Student_info)

学生0

学生1

学生2

学生3

学生4

学生5

.....

//以"读"方式打开文件

```
FILE *fp = fopen("students.info", "r");
```

```
if(fp == NULL) {  
    printf("打开文件失败!");  
    exit(0);  
}
```

//文件读写指针指向编号为5的学生记录

```
fseek(fp, 5*sizeof(Student_info), SEEK_SET);
```

```
Student_info stu;
```

//从文件读出1条记录，记录大小为 sizeof(Student_info)

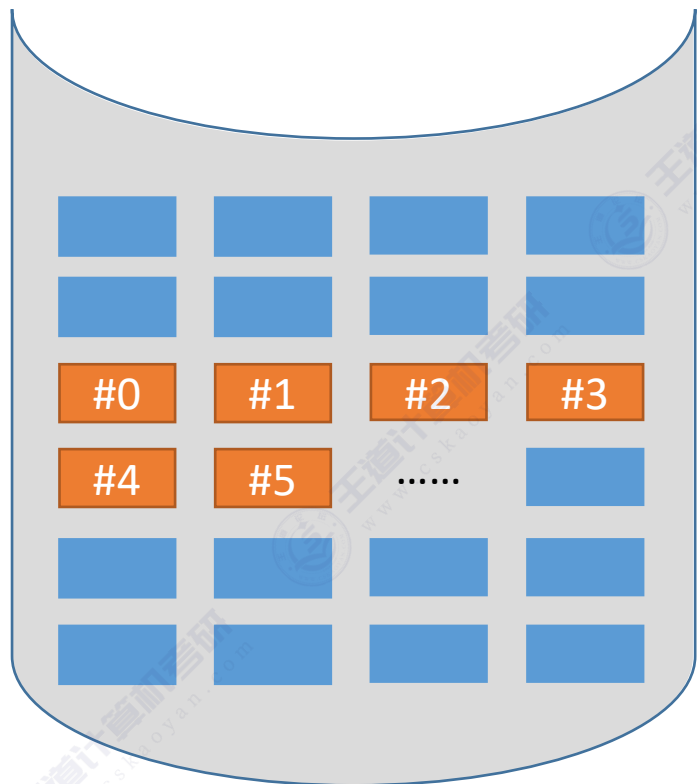
```
fread(&stu, sizeof(Student_info), 1, fp);
```

```
printf("学生编号: %d\n", stu.number);
```

```
fclose(fp);
```

用户用逻辑地址访问文件

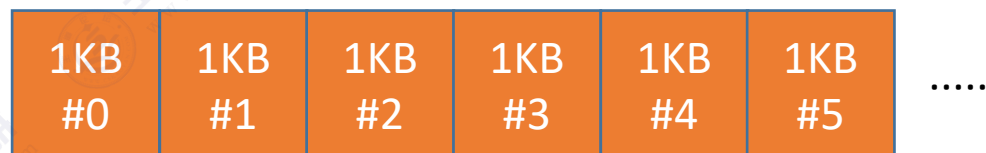
物理结构（从操作系统视角看）



用户视角：
每个学生记录占 64B
`sizeof(Student_info)`

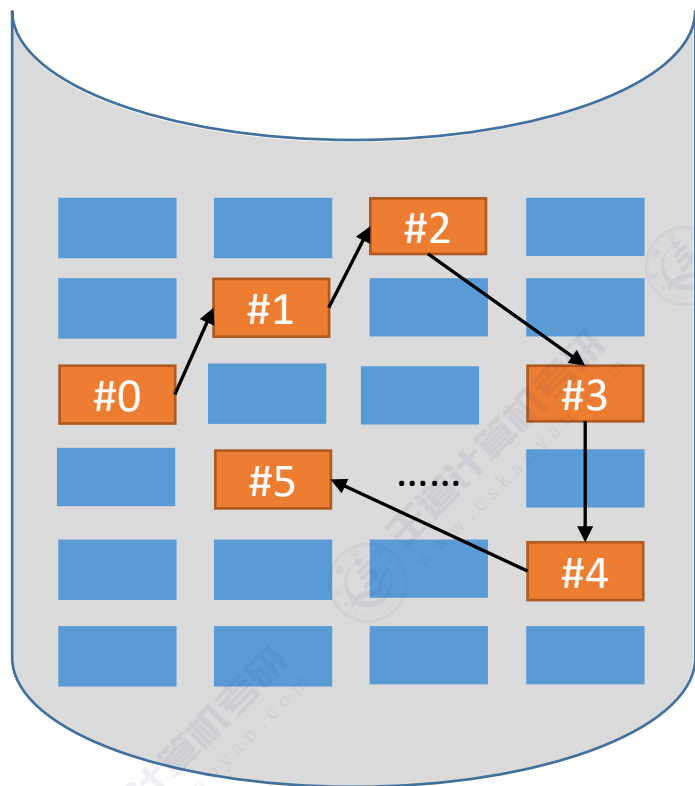


操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



连续分配：逻辑上相邻的块物理上也相邻

物理结构（从操作系统视角看）

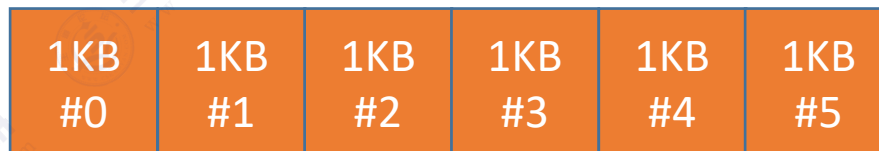


链接分配：逻辑上相邻的块在物理上用链接指针表示先后关系

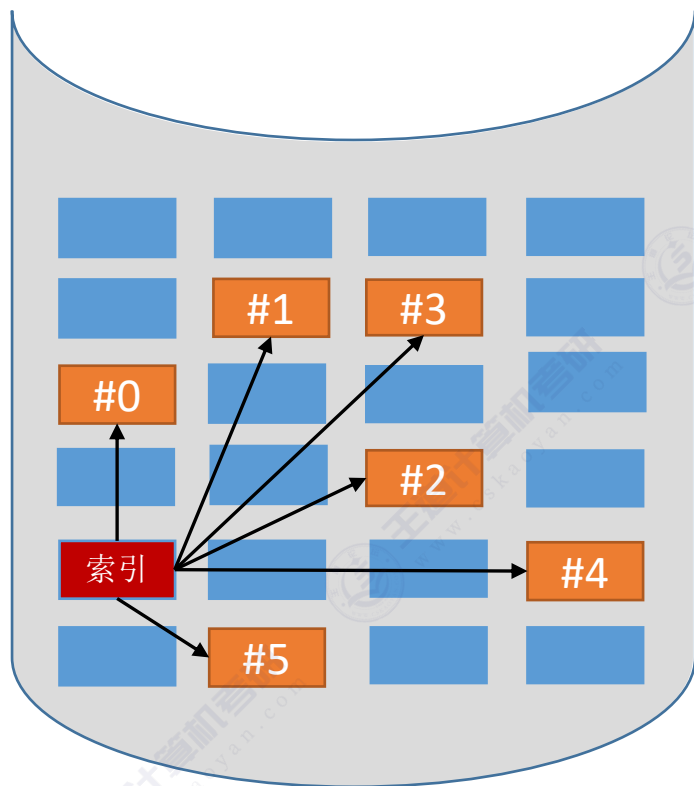
用户视角：
每个学生记录占 64B
`sizeof(Student_info)`



操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



物理结构（从操作系统视角看）

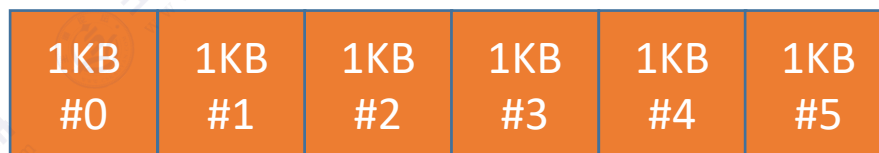


索引分配：操作系统为每个文件维护一张索引表，其中记录了逻辑块号→物理块号的映射关系

用户视角：
每个学生记录占 64B
`sizeof(Student_info)`



操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



懵逼点：顺序文件采用顺序存储/链式存储

顺序文件：各个记录可以顺序存储或链式存储。

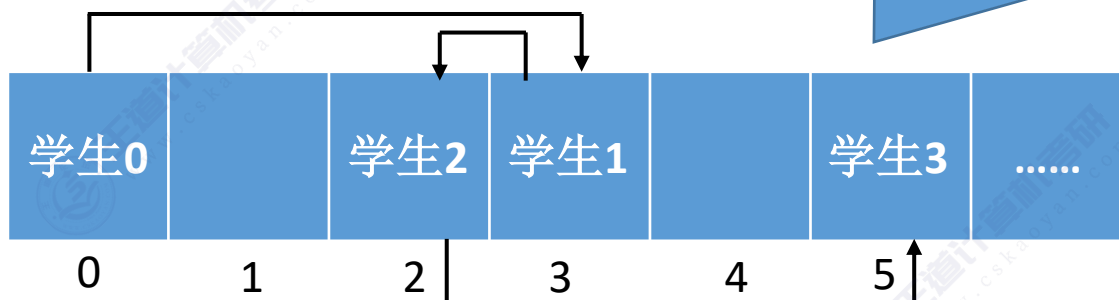
顺序存储，各条记录相邻这存放



支持随机访问：指可以直接确定第*i*条记录的逻辑地址

```
typedef struct {  
    int number;           //学号  
    char name[30];        //姓名  
    char major[30];       //专业  
} Student_info;
```

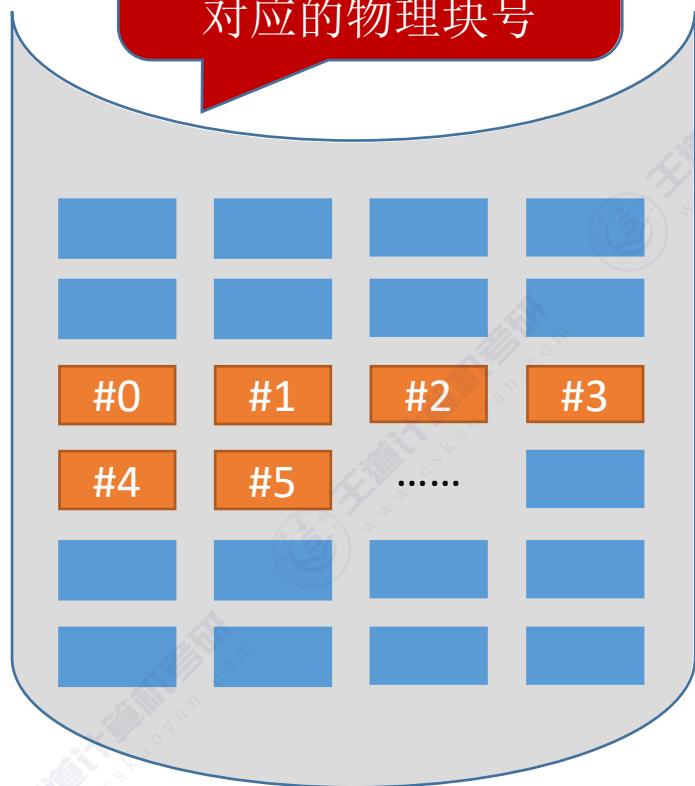
链式存储，各条记录离散着存放，用指针表示先后关系



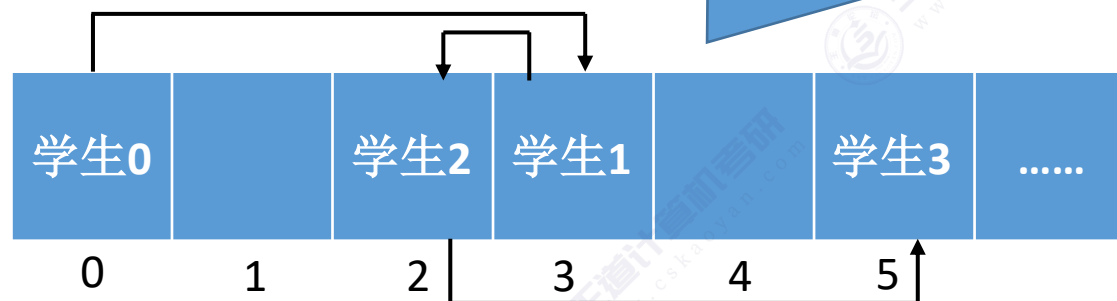
```
typedef struct {  
    int number;           //学号  
    char name[30];        //姓名  
    char major[30];       //专业  
    int next;             //下一个学生记录的存放位置  
} Student_info;
```

链式存储的顺序文件采用连续分配...

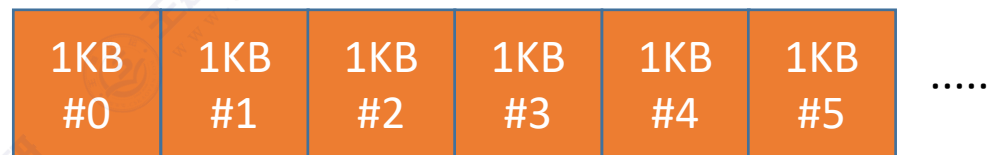
支持随机访问：指可以直接找到逻辑块号对应的物理块号



链式存储，各条记录离散着存放，用指针表示先后关系

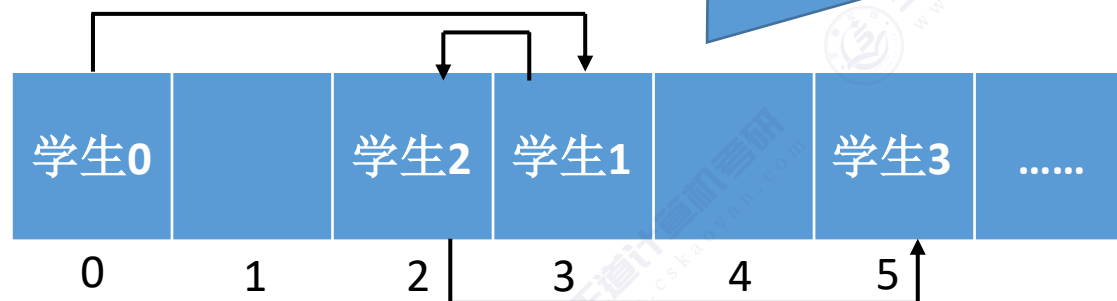
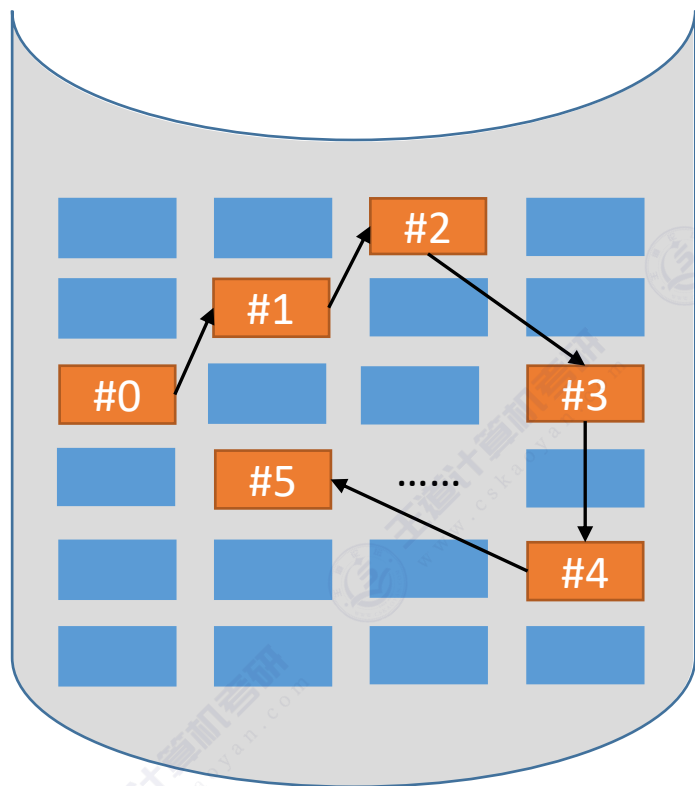


操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



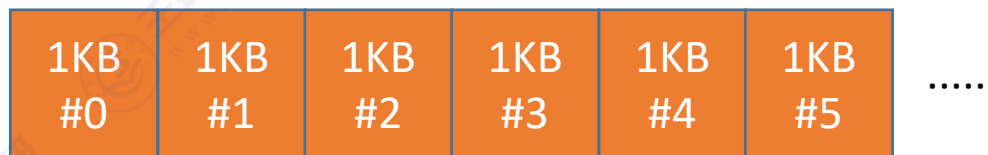
连续分配：逻辑上相邻的块物理上也相邻

链式存储的顺序文件采用链接分配...



链式存储，各条记录离散着存放，用指针表示先后关系

操作系统视角：反正就是一堆二进制数据，每个磁盘块可存储1KB，拆就完了！



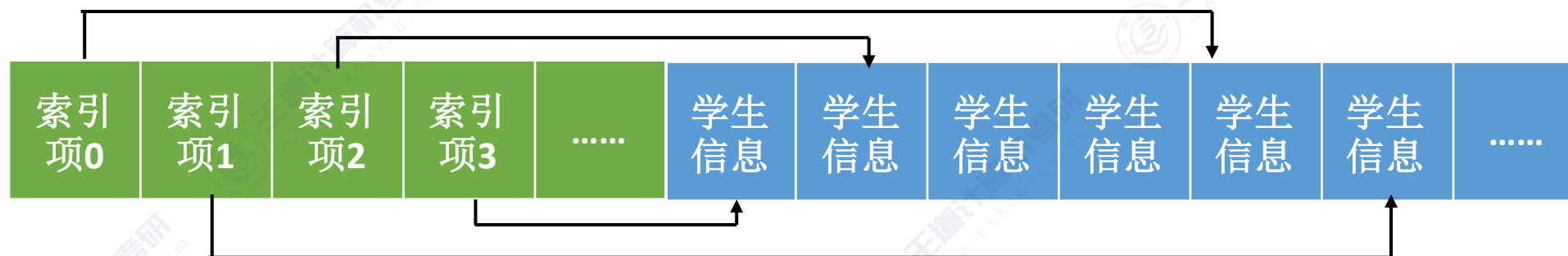
文件内部各条记录链式存储：由创建文件的用户自己设计的
文件整体用链接分配：由操作系统决定

关注公众号【研途小时】获取后续课程完整更新！

逻辑结构：索引文件

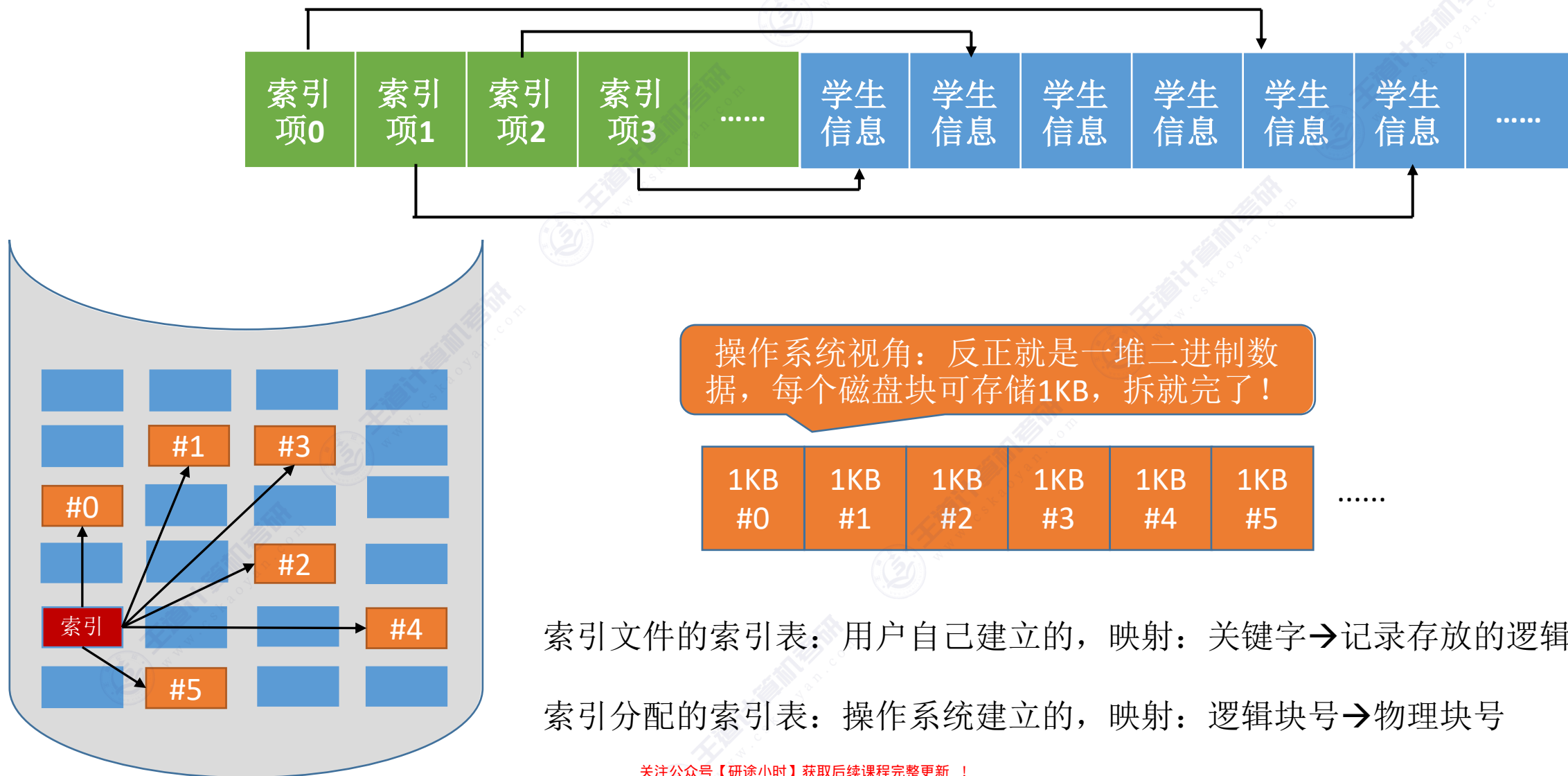
```
typedef struct {  
    int number;    //学号  
    int addr;      //学生记录的逻辑地址  
} IndexTable;
```

```
typedef struct {  
    char name[30];    //姓名  
    char major[30];   //专业  
    //还可添加其他各种各样的学生信息  
} Student_info;
```



索引文件：从用户视角来看，整个文件依然是连续存放的。如：前1MB存放索引项，后续部分存放记录。

索引文件采用索引分配...



索引文件的索引表：用户自己建立的，映射：关键字→记录存放的逻辑地址

索引分配的索引表：操作系统建立的，映射：逻辑块号→物理块号

慢下来消化一下8

逻辑结构 V.S. 物理结构

逻辑结构

用户（文件创建者）的视角看到的亚子

在用户看来，整个文件占用连续的逻辑地址空间

文件内部的信息组织完全由用户自己决定，操作系统并不关心

由操作系统决定文件采用什么物理结构存储

物理结构

操作系统负责将逻辑地址转变为（逻辑块号，块内偏移量）的形式，并负责实现逻辑块号到物理块号的映射



关注公众号【研途小时】获取后续课程完整更新！



@王道论坛



@王道计算机考研备考

@王道咸鱼老师-计算机考研

@王道楼楼老师-计算机考研



@王道计算机考研



等撩



等撩

知乎

@王道计算机考研

微信视频号

@王道计算机考研



微信公众平台

@王道在线



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研